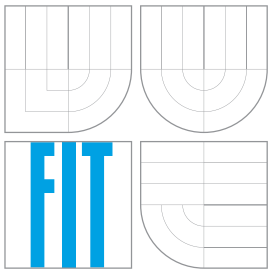
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYNTÉZA TEXTUR

TEXTURE SYNTHESIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

CTIRAD SOUSEDÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL

BRNO 2010

Abstrakt

Cílem práce je vytvořit co nejkvalitnější syntetizér textur a ukázkovou demonstrační utilitu s grafickým uživatelským rozhraním. Základem je metoda autorů Li-Yi Weie a Marca Lewoye ze Stanfordské univerzity, avšak jejich koncept je upraven se snahou dosáhnout lepších výsledků. Pro zkvalitnění syntézy je vypracována metoda detekce výrazných oblastí v textuře, které slouží při syntéze k inicializaci a zlepšují výsledek. Zde použité algoritmy nevyžadují větší pozornost a vstup od uživatele.

Abstract

This Bachelor's thesis deals with automatic texture synthesis. The main objective is to develop the best possible method of texture synthesis, inclusive a graphical demonstration utility. The original concept, developed by Li-Yi Wei and Marc Lewoy, Stanford University, has been modified to achieve better results. A method of detecting specific structures within texture images has been developed to initialize a synthesized image, and improve the final result. The employed algorithms are foolproof, and only limited numbers of input parameters have to be tuned.

Klíčová slova

textura, syntéza textur, Local Binary Patterns, počítačová grafika, počítačové vidění

Keywords

texture, texture synthesis, Local Binary Patterns, computer graphics, computer vision

Citace

Ctirad Sousedík: Syntéza textur, bakalářská práce, Brno, FIT VUT v Brně, 2010

Syntéza textur

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla.

.....
Ctirad Sousedík
18. května 2010

Poděkování

Děkuji panu Ing. Michalu Španělovi za jeho odbornou pomoc při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Syntéza textur je oblast, která se zabývá generováním textur zachovávajících očekávaný vzhled a strukturu. Výsledný vzhled a struktura může být zadána např. formou gramatiky, tento způsob je však použitelný obvykle pouze pro textury obsahující konkrétní opakující se objekty, u kterých je potřeba, aby byly předem známy. Daleko častěji je vzhled syntetizované textury zadán zdrojovou texturou, kterou chceme např. rozšířit na větší plochu, nebo potřebujeme opravit poškození v obraze texturou z okolí poškození.

Další oblastí, která spadá do syntézy textur, je syntéza temporálních textur, kdy textura je definována i v čase. Cílem je pak generovat např. na sebe navazující animaci takovéto temporální textury. Příkladem může být plápolající oheň nebo tekoucí voda.

Tato bakalářská práce je zaměřena na syntézu klasické textury z daného texturního vzoru. Cílem je pokusit se upravit některou z existujících metod syntézy textur tak, aby bylo dosaženo lepších výsledků co se týče vzhledu výsledné syntetizované textury. Je kladen velký důraz na stranovou návaznost (seamless texture) výsledku kvůli potenciálnímu použití v 3D grafice. Hlavní myšlenkou je obohatit některou ze současných metod syntézy textur o detekci určitých výrazných objektů, které hrají při subjektivním hodnocení výsledku největší roli. Pomocí detekce bude možné tyto objekty najít a lépe vyřešit jejich rozmístění v syntetizovaném výsledku. Výslednou metodu syntézy je pak cílem demonstrovat pomocí utility s grafickým uživatelským rozhraním.

V kapitole Teoretická část (2) jsou shrnuty existující metody a přístupy použité v této bakalářské práci. Kapitola Současný stav (3) shrnuje přístupy a metody, které v současné době pro syntézu textur existují. Jádrem práce je pak Návrh řešení (4) a Výsledky (6). V Návrhu řešení jsou popsány postupy a metody, které byly pro dosažení cíle použity. V Kapitole Výsledky (6) je pak vidět, jaké kvality výsledných syntetizovaných textur se podařilo dosáhnout.

Kapitola 2

Teoretická část

Texturu je obtížné definovat. Lze však říci, že se jedná o obraz splňující určité podmínky. Obvykle se u ní očekává, že bude zachycovat určitou strukturu. Pokud touto strukturou bude vyplněna libovolně velká plocha, bude takový obraz opět vnímán jako zobrazující stejnou strukturu. Z toho vyplývají určité vlastnosti, které jsou pro danou texturu specifické. Mezi tyto by bylo možné zařadit návaznost částí textury na jiné části textury, prostorové vztahy těchto částí a také vysokou pravděpodobnost jejich vzájemné podobnosti. Podle uvedených vztahů lze textury rozlišovat v různých stupních od regulárních po stochastické. Za regulární jsou považovány ty, v nichž jsou patrné jasné deterministické vztahy mezi objekty, a to jak prostorové, tak ty, které se týkají jejich vzájemné podobnosti. Příkladem může být pravidelná mřížka, šachovnice atd. Jako stochastické jsou naopak označovány textury, kde vzájemné vztahy objektů v textuře jsou obtížně definovatelné, náhodné a samotné rozeznání jednotlivých objektů je problematické. Příkladem může být koberec bez vzoru, omítka aj.

Problémem také je, že texturu je třeba chápat jako systém na různých úrovních rozlišení. Dříve zmíněné vztahy se vyskytují jak mezi malými tak mezi většími objekty v textuře. Větší objekty jsou pak často tvořeny skupinou menších objektů, což zachycení vzájemných vztahů dále komplikuje.

Syntéza textur tedy musí zahrnovat metody, které budou respektovat tyto vlastnosti, a využívat je pro generování cílového obrazu.

Existují přístupy, kdy jsou zmíněné vlastnosti definovány pomocí gramatik nebo jiných přesných pravidel. Syntéza pak generuje obraz podle těchto pravidel. Problémem uvedeného přístupu je, že nalezení takovýchto přesných pravidel je velmi obtížné, a je to možné pouze u textur, kde se vyskytují velmi jasně identifikovatelné opakující se objekty ve zřejmých prostorových vztazích. Tyto požadavky splňují obvykle jen regulární textury.

Proto je daleko častější přístup, kdy vzhled je zadán menší zdrojovou texturou a cílem je generovat texturu, která bude co nejpodobnější původní. Případně může být požadována navíc stranová návaznost syntetizované textury (seamless texture), aby mohla být použita ve 3D grafice. Metody využívající tento přístup lze rozdělit dále podle toho, zda dbají na vzájemnou podobnost okolí objektů v syntetizované a původní textuře pro každý pixel zvlášť, nebo pro větší výřezy obrazu. Druhá skupina se využíváním větších výřezů snaží snížit výpočetní náročnost a částečně také zachovat celistvost těchto výřezů v generované textuře.

2.1 Metody generující texturu z původní zadané textury

Tyto metody se snaží zajistit zachování vzájemných prostorových vztahů objektů v syntetizované textuře pomocí analýzy jejich okolí ve zdrojové textuře. Při samotné syntéze se pak snaží skládat cílový obraz, přičemž dbají na zachování co nejlepší podobnosti okolí bodů v syntetizované textuře s okolím bodů ve zdrojové textuře. Problém výskytu prostorových vztahů na různých úrovních rozlišení řeší obvykle buďto tak, že umožňují volbu na jaké úrovni rozlišení bude docházet ke zkoumání okolí, nebo využívají obrazových pyramid či jiných prostředků, jak analyzovat okolí na více úrovních rozlišení najednou.

2.2 Metoda Li-Yi Wei, Marc Levoy: Fast Texture Synthesis using Tree-sructured Vector Quantization [6]

Tato bakalářská práce vychází v oblasti samotné syntézy textury z metody autorů Li-Yi Weie a Marca Levoye ze Stanfordské univerzity. Jejich metoda patří do skupiny metod pracujících na úrovni jednotlivých pixelů co se týče zkoumání okolí. Jejím vstupem je původní textura a rozměry výsledné syntetizované textury. Je možné parametrizovat velikost zkoumaného okolí. Výstupem je pak výsledná syntetizovaná textura.

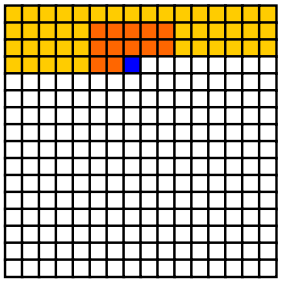
Pro popis funkce metody je nejprve vhodné popsat, jak pracuje na jediné úrovni rozlišení, a následně objasnit rozšíření tohoto konceptu pro více rozlišení najednou.

V rámci jednoúrovňové verze metody je vytvořen obraz cílových rozměrů, a je inicializován bílým šumem. Následně se obraz prochází pixel po pixelu, přičemž se vždy zkoumá, který pixel v původním obraze má okolí nejpodobnější s okolím v doposud vygenerovaném cílovém obraze. Takovýto pixel se zapíše na místo aktuálně zkoumaného pixelu v cílovém obraze.

Celý proces lze popsat následujícím pseudokódem:

```
funkce PixelyObrazku(obrazek)
{
  Vrať pixely obrázku od pravého horního rohu
  po řádích směrem k pravému dolnímu rohu.
}

funkce ZiskejOkoli(pixel)
{
  Vrať okolí pixelu.
}

funkce VratPixelSNejpodobnejsimOkolim(okoli, prohledavany obraz)
{
  Vrať pixel, jehož okolí je nejpodobnější zadanému okolí,
  v zadaném prohledávaném obraze.
}

funkce InicializujBilymSumem(obrazek)
{
  Inicializuje zadaný obrázek bílým šumem.
}
```

```
PuvodniObrazek = Původní textura ze které bude syntetizováno

GenerovanyObrazek = vytvoř cílový obraz s patřičnými rozměry
InicializujBilymSumem(GenerovanyObrazek)

pro každý pixel v PixelyObrazku(obrazek)
{
    OkoliVCilovemObraze = ZiskejOkoli(pixel)

    pixel = VratPixelSNejpodobnejsimOkolim(
                        OkoliVCilovemObraze,
                        PuvodniObrazek)
}
```



Obrázek 2.1: Procházení obrázkem

Důležitý je také tvar okolí, které se užívá k vyhledání nejvhodnějšího pixelu cílového obrazu. Mělo by být zvoleno tak, aby bylo ovlivněno původně inicializovaným bílým šumem v cílovém obraze pouze zpočátku, a poté už byla syntéza zcela deterministická. V opačném případě budou zvolená okolí příliš náhodná a metoda nebude fungovat dle očekávání.

Vhodné okolí lze definovat následujícím způsobem:

$$O_1 = \{(x,y) : x \in X_1 \wedge y \in Y_1\} \cup \{(x,y) : x \in X_2 \wedge y \in Y_2\} \tag{2.1}$$

kde

$$X_1 = \{x : x \geq x_p - s \wedge x \leq x_p + s \wedge x \in X_o\} \tag{2.2}$$
$$Y_1 = \{y : y \geq y_p - s \wedge y < y_p \wedge y \in Y_o\} \tag{2.3}$$
$$X_2 = \{x : x \geq x_p - s \wedge x < x_p \wedge x \in X_o\} \tag{2.4}$$
$$Y_2 = \{y : y = y_p \wedge y \in Y_o\} \tag{2.5}$$

a kde

X_o	je množina indexů od 0 do (šířky obrázku - 1)
Y_o	je množina indexů od 0 do (výšky obrázku - 1)
s	je úroveň velikosti okolí
x_p	je index x, pixelu jehož okolí určujeme
y_p	je index y, pixelu jehož okolí určujeme

Takovéto okolí bude dále nazýváno jako půlčtvercové (2.1), jeho tvar pro $s = 2$ a způsob procházení obrázkem demonstruje také obrázek 2.1.

Další okolí použité v této bakalářské práci je definováno jako

$$O_2 = \{(x,y) : x \in X_1 \wedge y \in Y_1\} - \{(x,y) : x = x_p \wedge y = y_p\} \tag{2.6}$$

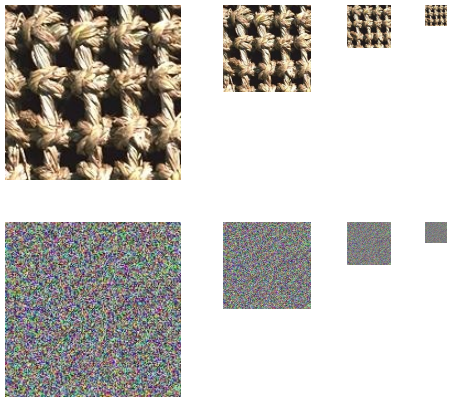
kde

$$X_1 = \{x : x \geq x_p - s \ \wedge x \leq x_p + s \ \wedge x \in X_o\} \tag{2.7}$$
$$Y_1 = \{y : x \geq y_p - s \ \wedge y \leq y_p + s \ \wedge y \in Y_o\} \tag{2.8}$$

kde

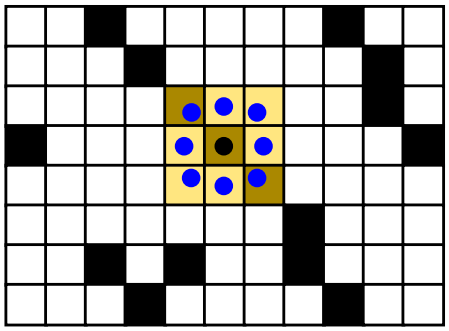
X_o	je množina indexů od 0 do (šířky obrázku - 1)
Y_o	je množina indexů od 0 do (výšky obrázku - 1)
s	je úroveň velikosti okolí
x_p	je index x pixelu, jehož okolí určujeme
y_p	je index y pixelu, jehož okolí určujeme

Takovéto okolí bude dále označováno jako čtvercové (2.6).



Obrázek 2.2: Generovaná obrazová pyramida [7]

Tato metoda však pracuje zároveň na více úrovních rozlišení, a to následujícím způsobem. Z původního obrazu je vygenerována obrazová pyramida, kde každá úroveň je 2× zmenšeným obrazem úrovně o jednu vyšší (viz. obrázek 2.2). Stejným způsobem je obrazová pyramida generována i z cílového obrazu inicializovaného na zadané rozměry bílým šumem. Syntéza je pak prováděna dříve zmíněným způsobem nejprve na vyšších úrovních pyramidy, a teprve následně na těch nižších. Podstatné je, že při hledání vhodného pixelu, který má být umístěn do generované textury, se neuvažuje pouze okolí na dané úrovni pyramidy, ale také na úrovni o jednu výše. Tímto způsobem se nejprve syntetizuje výsledek ze zmenšené verze zdrojové textury do zmenšené verze cílové textury, a tento pak dále ovlivňuje syntézu vyšších rozlišení. Výsledek tak lépe zahrnuje vztahy mezi objekty v textuře na nižších úrovních rozlišení, aniž by bylo nutné zvětšovat zkoumané okolí. V podstatě se tak nejprve generují do cílového obrazu větší objekty, které jsou rozlišitelné dobře i ve vyšších úrovních pyramidy, a teprve následně jsou v nižších úrovních syntetizovány jejich detaily. Tento přístup výrazně snižuje výpočetní náročnost zachycení vztahů větších objektů v textuře.



Obrázek 2.3: okolí u *Local Binary Patterns*

2.3 Local Binary Patterns [5]

V této bakalářské práci jsou použity i některé postupy vycházející z metod počítačového vidění. Konkrétně byla využita metoda pro klasifikaci textur *Local Binary Patterns*. Metoda zkoumá okolí jednotlivých pixelů v zadané oblasti a ze získaných výsledků vytváří histogram. V klasické podobě se analyzuje vztah intenzity středového pixelu a jednotlivých pixelů bezprostředně sousedících s tímto pixelem (viz obrázek 2.3). Podle toho, zda je intenzita jednotlivých bezprostředně sousedících pixelů větší nebo menší, se jako výsledek srovnání generuje 0 nebo 1. Vzhledem k tomu, že bezprostřední okolí pixelu tvoří jiných 8 pixelů, je takto možné vytvářet osmibitové číslo. Vhodným způsobem klasifikace textur je pak zkoumání histogramů četnosti výskytu jednotlivých osmibitových kombinací v daném výřezu textury. Metodu je možné upravit do tzv. uniformních *Local Binary Patterns*, kdy se z výsledného histogramu vyřazují určité osmibitové kombinace. Ukázalo se totiž, že některé kombinace, obzvláště pak ty, které se vyznačují častými změnami hodnot 1 a 0, nejsou pro klasifikaci příliš užitečné, jelikož popisují především šum v obraze.

Kapitola 3

Současný stav

Syntéza textur je oblast, která se zabývá generováním textur zachovávajících očekávaný vzhled a strukturu. Výsledný vzhled a struktura může být zadána např. formou gramatiky, tento způsob je však použitelný obvykle pouze pro textury obsahující konkrétní opakující se objekty, u kterých je potřeba, aby byly předem známy. Daleko častěji je vzhled syntetizované textury zadán zdrojovou texturou, kterou chceme např. rozšířit na větší plochu, nebo potřebujeme opravit poškození v obraze texturou z okolí poškození. Další oblastí, která spadá do syntézy textur, je syntéza temporálních textur, kdy textura je definována i v čase. Cílem je pak generovat např. na sebe navazující animaci takovéto temporální textury. Příkladem může být plápolající oheň nebo tekoucí voda.

Tato bakalářská práce je zaměřena na syntézu klasické textury z daného texturního vzoru.

3.1 Rozdělení metod syntézy textur

Metody syntézy textur ze zadaného texturního vzoru lze rozdělit do dvou základních skupin podle přístupu.

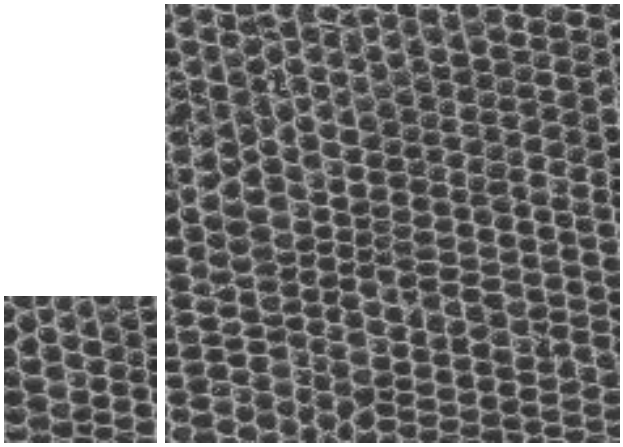
První skupina zahrnuje metody, které textury generují tak, že jako každý generovaný pixel se užije pixel ve zdrojové textuře, jehož okolí nejlépe odpovídá okolí v cílové textuře. Metody často pracují podle schématu, ve kterém je nejprve cílový obraz určitým způsobem inicializován, a následně je metodou postupně transformován, pixel po pixelu, do cílové podoby syntetizované textury. Metody se obvykle liší způsobem procházení obrazem, zvolenými způsoby inicializace a funkcí, která určuje příslušnost toho kterého pixelu na danou pozici v syntetizované textuře. Tato funkce obvykle zkoumá okolí pixelu, který má být generován, a hledá pixel ve zdrojové textuře, který nejlépe vyhovuje zachování struktury a vzhledu textury.

Druhá skupina metod se pak vyznačuje tím, že místo s pixely pracuje s většími oblastmi v obraze. Tyto metody pracují s pevně zvolenými nebo proměnlivými výřezy zdrojového obrazu, které způsobem specifickým pro danou metodu skládají v cílovou syntetizovanou texturu. Díky tomu, že metody pracují nikoliv s pixely, ale jejich skupinami, mohou dosahovat potenciálně nižší výpočetní náročnosti.

3.2 Metody využívající Markov Random Fields

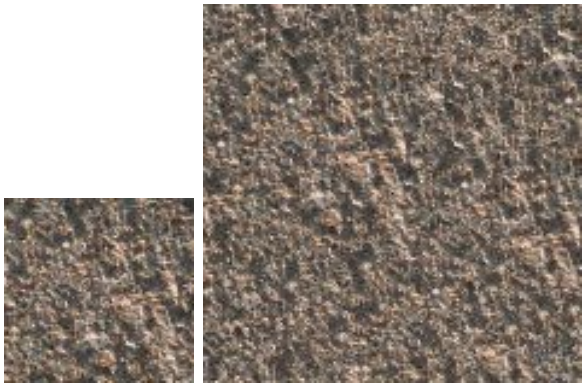
Zahrnují např. metodu Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field [4]. Metody užívají model Markov Random Fields přímo pro generování cílové textury. Dle svých autorů jsou úspěšné v generování širokého spektra textur, problémem je však jejich vysoká výpočetní náročnost.

3.3 Texture Synthesis by Non-parametric Sampling [2]



(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.1:

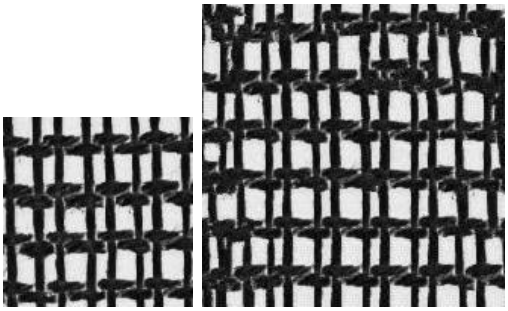


(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.2:

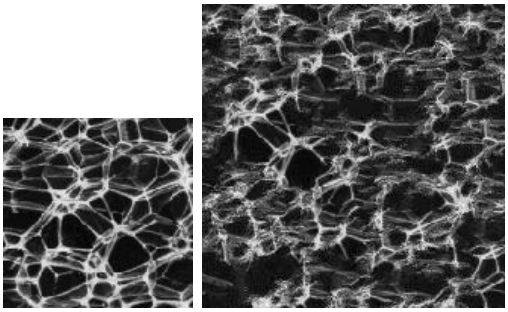
Metoda generuje obrázek v výchozího inicializačního bodu, pixel po pixelu. Původní obraz je dotazován na okolí podobné okolí pixelu v již syntetizovaném obraze. Pixel s vhodným okolím je vybrán do syntetizovaného obrazu. Lze regulovat úroveň náhodnosti procesu výběru okolí.

3.4 Fast Texture Synthesis using Tree-sructured Vector Quantization [6]



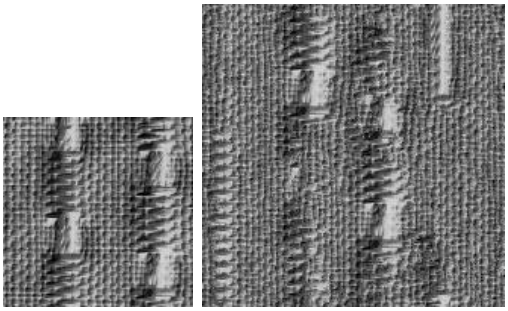
(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.3:



(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.4:



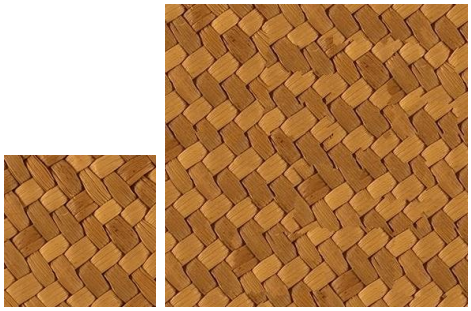
(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.5:

Obraz, do kterého má být syntetizována textura, je inicializován bílým šumem, a následně

je generována textura pixel po pixelu. Každý pixel se vybere z původního obrazu, a to na základě podobnosti mezi okolím daného pixelu v původním obraze a okolím pixelu v generovaném obraze. Metoda navíc užívá při výběru pixelu místo jediného původního obrazu obrazovou pyramidu generovanou z původního obrazu. Výsledkem je lepší zachování větších struktur v syntetizované textuře. Výpočetní náročnost je dle autorů výrazně nižší než u metody Texture Synthesis by Non-parametric Sampling.

3.5 Image Quilting [3]



(a) zdrojová textura (b) syntetizovaná textura

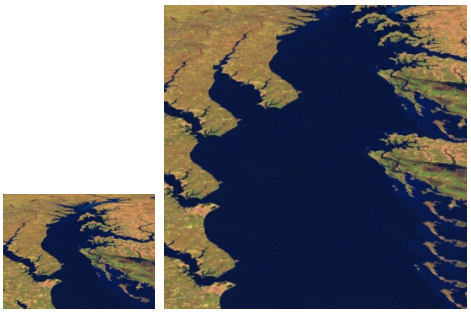
Obrázek 3.6:

(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.7:

Patří mezi metody pracující s většími výřezy obrazu. Oproti pixelovým metodám vykazuje výrazně nižší výpočetní náročnost. Metoda skládá výřezy obrazu přes sebe tak, aby na sebe dobře navazovaly, a tímto způsobem syntetizuje výslednou texturu.

3.6 Image and Video Synthesis Using Graph Cuts [1]



(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.8:

(a) zdrojová textura (b) syntetizovaná textura

Obrázek 3.9:

Metoda rovněž pracuje s většími výřezy obrazu, ty jsou však nepravidelné a jsou identifikovány podle analyzovaného texturního vzoru. Cílem je vybírat oblasti, které by i v generované textuře měly být jednolité, a teprve z nich generovat výsledek. Vyžaduje kromě algoritmu syntézy také detekci těchto oblastí. Metoda pracuje iterativně a snaží se najít dobré rozložení identifikovaných výřezu v syntetizované textuře, nezabývá se však stranovou návazností výsledku. Autoři uvádějí její použitelnost i pro syntézu temporálních textur.

Kapitola 4

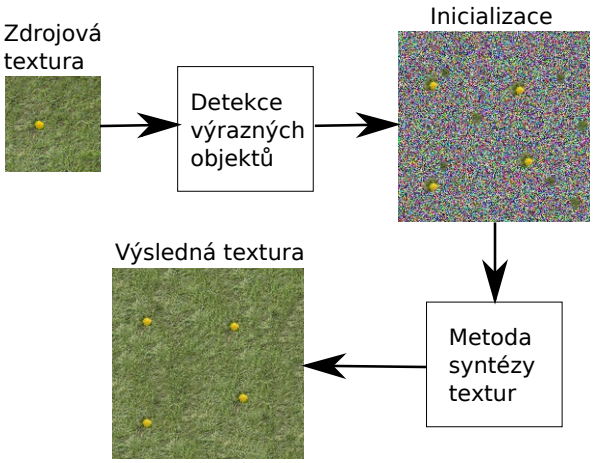
Návrh řešení

Vzhledem k tomu, že utilita má být potenciálně prakticky použitelná, bylo vhodné uvážit aspekty, které toto obnáší.

Obvyklým vstupem syntézy bude obrázek, u kterého se předpokládá, že bude splňovat určité vlastnosti textury. Lze však také očekávat, že vzor nebude pro účely syntézy dokonale upraven. Lze především předpokládat, že v textuře se budou vyskytovat objekty, které budou velmi výrazné, a při přímé aplikaci metod syntézy textur bude docházet k nevhodným jevům.

Ve zdrojových texturách se mohou vyskytovat rozličné kazy a výrazné objekty, např. suk ve dřevě, květina v trávě nebo škrábanec v omítce zdi. Tyto objekty obvykle očekáváme v cílové generovaně textuře, avšak rozmístěné náhodně tak, aby se vyskytovaly přibližně stejně často jako ve zdrojové textuře. Vzhledem k tomu, že výrazné objekty se vyskytují v textuře ojediněle, mohou klasické metody syntézy textur tyto objekty z generování zcela vyřadit nebo naopak je v obraze generovat častěji, než je očekáváno.

Výše zmíněný problém je v této práci řešen pomocí metod počítačového vidění. Cílem je výrazné objekty ve zdrojové textuře detekovat, rozmístit je na plochu cílové generované textury tak, aby byl dosažen požadovaný vzhled, a teprve následně využít metod syntézy textur ke generování obrazu. Samotná syntéza pak bude ovlivněna touto externí inicializací, a mělo by se podařit dosáhnout lepších výsledků.



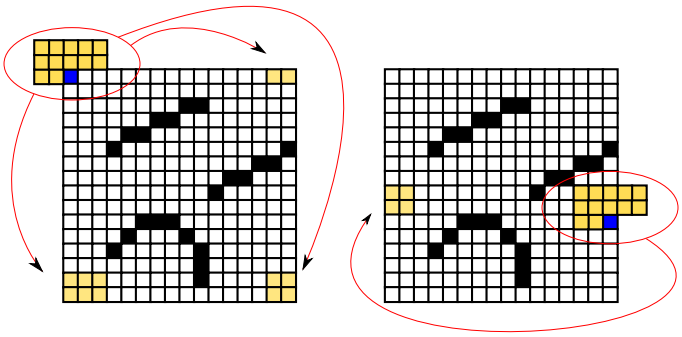
Obrázek 4.1: blokové schéma řešení.

Po prozkoumání, jaké možnosti a výsledky jednotlivé metody syntézy textur poskytují a jaké mají silné a slabé stránky, byla vybrána jako primární metoda pro syntézu metoda Fast Texture Synthesis using Tree-sructured Vector Quantization [6] autorů Li-Yi Weie a Marca Levoye. Bude však vhodné vyjít pouze z jejich hlavních konceptů a experimentováním se pokusit dosáhnout co nejlepších výsledků, a to především proto, že metoda nepočítá s plánovanou inicializací detekovanými výraznými objekty.

Při řešení bylo nejprve přistoupeno k experimentování s metodou syntézy textur. Cílem bylo vytvořit implementaci metody, která bude schopná na základě ručně připravených inicializací výraznými objekty, syntetizovat co nejlepší výslednou texturu.

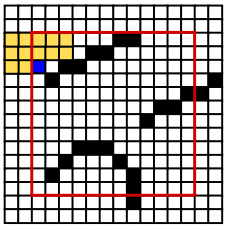
4.1 Syntéza textury pomocí metody Li-Yi Weie a Marca Lewoye

4.1.1 Stranová návaznost textury



Obrázek 4.2: přesahující okolí v obraze

U vstupní textury nelze očekávat, že bude sama na sebe stranově navazovat (seamless). Naopak by výsledná utilita měla být schopna generovat stranově navazující texturu ze stranově nenavazujícího vzoru. Ve zdrojové textuře tedy lze počítat pouze s návazností v rámci obrazu samotného. Okolí, pomocí kterého je zkoumán zdrojový obraz má však rozměry, které u okrajových pixelů zdrojové textury budou přesahovat její rozměry. Vzhledem k tomu, že jak již bylo řečeno nelze očekávat stranovou návaznost zdrojového textury, není možné užít klasický přístup, kdy by se přesahující pozice pixelů okolí modulovaly pomocí rozměrů textury, jak ukazuje obrázek 4.2.



Obrázek 4.3: výřez zdrojové textury užitý k hledání pixelů s vhodným okolím

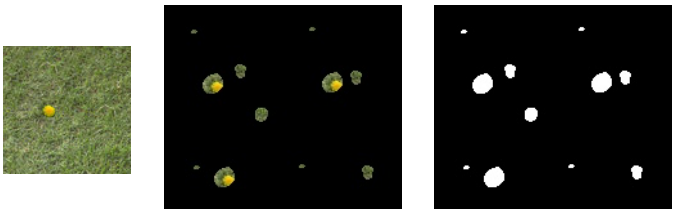
Znamená to totiž, že se z hlediska okolí počítá se stranovou návazností zdrojové textury, a úprava metody tímto způsobem skutečně generovala špatné výsledky. Řešením je prohle-

dávat zdrojovou texturu tak, aby i zvolené okolí zkoumaných pixelů nepřesahovalo hranice obrázku, jak je vidět na obrázku 4.3. Zvolený přístup má však tu nevýhodu, že pixely na okrajích zdrojové textury nebudou nikdy použity do generovaného obrazu. Malé objekty na okrajích zdrojové textury tak nebudou přeneseny do syntetizované textury. Vzhledem k povaze textury to však nehraje velkou roli, protože ojedinělý výskyt malých objektů pouze na okrajích je pro texturu zcela netypický.

Dále bylo potřeba zabývat se tím, jak zajistit stranovou návaznost syntetizované textury samotné. Zvolená metoda syntézy je pro daný účel částečně uzpůsobena, přesto však bylo třeba vyřešit určité problémy. Stranová návaznost levé strany na pravou u syntetizované textury nepředstavovala problém, protože je vedlejším důsledkem metody. U generovaného obrazu je totiž okolí analyzováno u okrajových pixelů tak, že přesahující části okolí se berou z druhé strany obrazu, pozice pixelů okolí jsou tedy modulovány rozměry obrázku. Díky tomu syntéza pixelů na pravých okrajích obrázku zahrnuje do okolí již generované pixely na straně levé, a vytváří tak návaznost. O něco problematičtější je návaznost textury samy na sebe shora a zdola. Zde totiž tvar okolí nic takového nezajišťuje. Nabízí se myšlenka užívat zcela čtvercové okolí, které by daný účel plnilo. Není to však možné, protože je pak při generování zahrnuto příliš mnoho pixelů stále ještě inicializovaných bílým šumem a metoda přestává fungovat. Vhodným řešením bylo syntetizovat většinu obrazu klasicky, a zvlášť se zabývat pruhem obrázku v dolní části, kde by byla v případě plně čtvercového (2.6) okolí generována návaznost. Tento pruh je syntetizován zvlášť totožným postupem jako zbývající část obrazu, je však použito okolí plně čtvercového tvaru, kde střed tohoto čtverce tvoří v dané chvíli syntetizovaný pixel.

4.1.2 Inicializace

Pokud je syntéza prováděna s cílovým obrazem inicializovaným pouze jako bílý šum, má především pruh šířky s (viz (2.1)) v horní části obrázku tendenci být generován jako plocha, jejíž vzhled dobře neodpovídá zdrojové textuře. Tento problém patrně způsobuje fakt, že okolí nejpodobnější inicializovanému náhodnému šumu mají tendenci být příliš náhodně vybírána z celé zdrojové textury. Zvolené řešení spočívá v inicializaci cílového obrazu částí obrazu zdrojového. Problematický pruh obrázku je tak ovlivněn inicializací původním obrazem, a daleko lépe odpovídá zdrojové textuře. Dále bylo nutné vyřešit, jakým způsobem

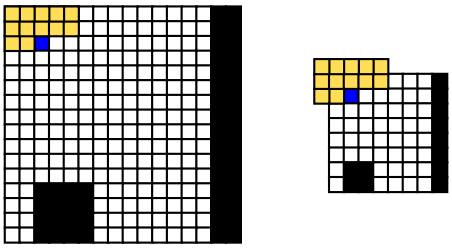


Obrázek 4.4: zdrojová textura, inicializace bez bílého šumu, maska inicializace

umožnit inicializaci syntetizované textury detekovanými výraznými objekty. Experimenty ukázaly, že použitelné je rozmístit objekty po ploše textury a pixely, které tyto objekty zaplňují, označit také v binární masce, jak ukazuje obrázek 4.4. Samotný proces syntézy pak nepřepisuje takto inicializované pixely, ale zároveň jejich hodnoty využívá při výběru vhodného okolí. Detekované objekty jsou tak zagenerovány do struktury syntetizované textury. Do jednotlivých úrovní pyramidy jsou pak generovány zmenšené verze inicializace cílové textury, a také masky této inicializace.

4.1.3 Syntéza pomocí obrazové pyramidy

Syntéza textury probíhá pomocí obrazové pyramidy zároveň na více úrovních. Bylo potřeba vyřešit, jak ideálně při syntéze obrazu v nižších úrovních pyramidy zahrnout již syntetizovaný obraz z úrovně vyšší, aby bylo dosaženo co nejlepšího výsledku. Okolí na aktuální syntetizované úrovni je půlčtvercové (2.1). Při použití půlčtvercového okolí také pro vyšší úroveň metoda sice vykazovala přijatelné výsledky, měla však tendenci u pravidelných objektů v obraze, u kterých očekáváme pravidelný tvar i v syntetizované textuře, vytvářet nevhodné útvary. Jednalo se natahování ve vertikálním směru, které patrně souvisí s tím, že půlčtvercové okolí zahrnuje pouze pixely nad právě syntetizovaným pixelem, ale už nikoliv pod ním. Pomocí experimentu bylo ale zjištěno, že použitím čtvercového okolí (2.6) ve vyšší úrovni pyramidy lze dosáhnout daleko lepších výsledků. Metoda pak měla lepší tendenci přizpůsobovat právě syntetizovaný obraz již syntetizovanému obrazu na vyšší úrovni pyramidy. Bylo nutné také vyřešit problém související s rozdílnými velikostmi úrovní pyramidy.



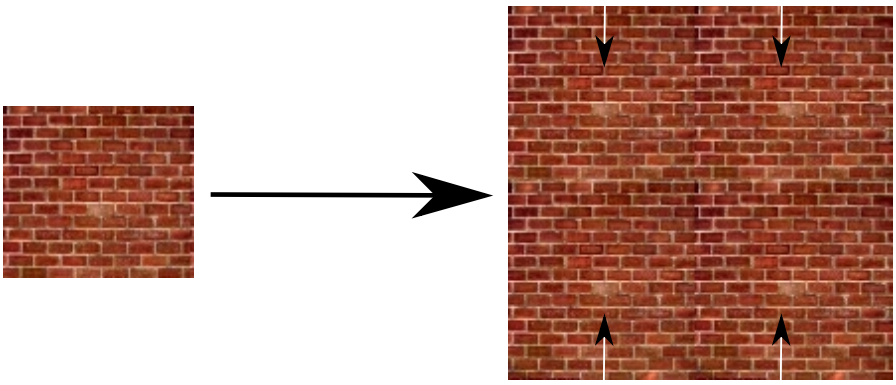
Obrázek 4.5: výřez zdrojové textury užitý k hledání pixelů s vhodným okolím

Ve vyšší úrovni pyramidy je použito okolí stejné velikosti, obrázek je však 2x menší. To znamená, že okrajové pixely v nižší úrovni zdrojové textury, jejichž okolí nepřesahuje okraje textury, by ve vyšší úrovni mohly odpovídat pixelům, jejichž okolí již okraje přesahuje. Podstatu problému demonstruje obrázek 4.5. Použité řešení spočívá v tom, že jsou na nižší úrovni obrazové pyramidy k syntéze použity jen ty pixely, které na vyšší úrovni nezpůsobí přesah okolí přes okraje obrazu. Sice se takto dále redukuje oblast, která je ze zdrojové textury využita k syntéze cílové textury, avšak získané výsledky svědčily pro toto řešení.

4.2 Detekce výrazných oblastí v obraze

Cílem bylo vytvořit postup, kterým bude možné za pomocí metod počítačového vidění extrahovat ze zdrojové textury výrazné objekty. Pokud by se totiž takovéto objekty v syntetizované textuře vyskytovaly častěji, jinak prostorově rozložené než ve zdrojové textuře, či by byl výrazně změněn jejich tvar, bylo by to lidským okem okamžitě vnímáno jako selhání syntézy. Je ovšem problematické stanovit, co přesně označit jako výrazný objekt. Na první pohled ve zdrojové textuře nemusí být patrný výskyt žádných výrazných objektů, avšak pokud je texturou vyplněna větší plocha, okamžitě začne být subjektivně zřejmé opakování a pravidelné rozložení objektů v takovém obraze - viz obrázek 4.6. Jedná se tedy o části textury, které se nejvýrazněji liší od struktur, které se obvykle v textuře vyskytují. Bylo by tedy vhodné sestavit funkci, která by byla schopná porovnat dvě oblasti v obraze, a stanovit míru této odlišnosti. Toto však může být problematické, jelikož jde o co nejlepší napodobení subjektivního vnímaní lidským zrakem.

Pomocí zmíněné funkce by pak bylo možné zkoumat zdrojovou texturu a hledat vhodným postupem extrémy funkce odlišnosti okolí. Při hledání vhodné funkce i způsobu detekce



Obrázek 4.6: výrazná místa v obraze

jejich extrémů byly vyzkoušeny následující postupy.

4.2.1 Detekční funkce

Při hledání vhodné funkce byl nejprve užíván obraz v odstínech šedi, aby byl každý pixel reprezentován pouze svým jasem. Výrazná místa a objekty se pak mohou lišit např. strukturou, celkovým jasem a histogramem jasu. Byly tedy s různou úspěšností vyzkoušeny různé detekční funkce, výsledky detekce jsou shrnuty v kapitole Výsledky (6).

Pro vystižení rozdílů v jasu mezi okolími byl vyzkoušen histogram světlosti pixelů, kde je v případě výskytu hodnoty dané třídy jasu přičítána hrubá aproximace Gaussovy funkce. Výhoda oproti přičítání klasického jednotkového skoku je ve vyšší podobnosti histogramů, kde byla světlost sledovaných pixelů v obraze podobná.

Použitý histogram světlosti pixelů $H[n]$ je definován následujícím způsobem:

$$H[n] = \sum_{i=0}^{C-1} G[n - p_i] \tag{4.1}$$

kde

C	je počet pixelů ve zkoumané oblasti
p_i	je hodnota jasu pixelu s indexem i, a to v rozsahu 0 až 255

$$G[x] = \left\{ \begin{array}{lll} 2 & \text{pro} & x = -2 \\ 8 & \text{pro} & x = -1 \\ 10 & \text{pro} & x = 0 \\ 8 & \text{pro} & x = 1 \\ 2 & \text{pro} & x = 2 \\ 0 & \text{pro} & x < -2 \vee x > 2 \end{array} \right\}$$

Koeficient odlišnosti d dvou takovýchto histogramů H_1, H_2 je pak

$$d = \sum_{i=0}^{255} |H_1[i] - H_2[i]| \tag{4.2}$$

Dále byl použit histogram klasických *Local Binary Patterns*, který počítá množství výskytů jednotlivých osmibitových čísel v definované oblasti. Tato čísla jsou sestavena na základě vztahu mezi jasem středového pixelu a okolních osmi pixelů.

Tento histogram je definován jako:

$$H[n] = \sum_{i=0}^{C-1} \delta[n - F_{LBP}[x_i, y_i]] \tag{4.3}$$

kde

C	je počet pixelů ve zkoumané oblasti
$F_{LBP}[x, y]$	je hodnota *LBP* příznaku v pixelu (x, y)
x_i	je x-ová souřadnice pixelu, který má ve zkoumané oblasti index i
y_i	je y-ová souřadnice pixelu, který má ve zkoumané oblasti index i

$$\delta[n] = \begin{Bmatrix} 1 & \text{pro} & n = 0 \\ 0 & \text{pro} & n \neq 0 \end{Bmatrix}$$

Koeficient odlišnosti d dvou takovýchto histogramů H_1, H_2 je

$$d = \sum_{i=0}^{255} |H_1[i] - H_2[i]| \tag{4.4}$$

Klasické *Local Binary Patterns* jsou však relativně odolné vůči změnám jasu a kontrastu zkoumané části obrazu. Středový pixel je u *LBP* porovnáván s osmi okrajovými pouze tak, že se stanovuje binární vztah 1 nebo 0 podle toho, zda je jas středového pixelu vyšší či nižší než jas okrajového. Celkové zvýšení kontrastu obrazu takovéto vztahy naruší velmi málo, stejně tak jako celkové zvýšení jasu. Nabízí se tedy vyzkoušet úpravu *Local Binary Patterns* tak, aby vztah mezi středovým pixelem a osmi okrajovými nebyl stanovován striktně jako 1 či 0, ale byl odstupňován jemněji. Jako další možnost se nabízí sledování jasu středového pixelu.

Z tohoto důvodu byla vyzkoušena úprava *Local Binary Patterns*, kde se místo pouhého histogramu počtu výskytů jednotlivých osmibitových čísel sleduje součet hodnot jasu středového pixelu histogramem H:

$$H[n] = \sum_{i=0}^{C-1} P[x_i, y_i] \cdot \delta[n - F_{LBP}[x_i, y_i]] \tag{4.5}$$

kde

C	je počet pixelů ve zkoumané oblasti
$F_{LBP}[x, y]$	je hodnota *LBP* příznaku v pixelu (x, y)
x_i	je x-ová souřadnice pixelu, který má ve zkoumané oblasti index i
y_i	je y-ová souřadnice pixelu, který má ve zkoumané oblasti index i
$P[x, y]$	je hodnota jasu pixelu (x, y)

$$\delta[n] = \begin{cases} 1 & \text{pro} \quad n = 0 \\ 0 & \text{pro} \quad n \neq 0 \end{cases}$$

Koeficient odlišnosti d dvou takovýchto histogramů H_1, H_2 je

$$d = \sum_{i=0}^{255} |H_1[i] - H_2[i]| \tag{4.6}$$

Byla také zkoušena funkce sledující přímo vztahy mezi středovým a okrajovými pixely, a to tak, že kromě stanovení samotného *LBP* příznaku je určen vektor osmi hodnot, který popisuje odlišnost jasu středového pixelu a jasu pixelů v jeho okolí. Tento vektor je pak přičítán k vektoru na jedné z 256 pozic histogramu. Pozice se určuje podle klasického *LBP* příznaku. Cílem bylo sledovat kromě samotných příznaků také vztahy mezi jasy porovnávaných pixelů. Způsob výpočtu vyjadřuje následující definice:

$$H[n, m] = \sum_{i=0}^{C-1} \sum_{j=0}^{7} (P[x_i, y_i] - P_O[x_i, y_i, j]) \cdot \delta[n - F_{LBP}[x_i, y_i], m - j] \tag{4.7}$$

kde:

C	je počet pixelů ve zkoumané oblasti
$F_{LBP}[x, y]$	je hodnota *LBP* příznaku v pixelu (x, y)
x_i	je x-ová souřadnice pixelu, který má ve zkoumané oblasti index i
y_i	je y-ová souřadnice pixelu, který má ve zkoumané oblasti index i
$P[x, y]$	je hodnota jasu pixelu (x, y)
$P_O[x, y, j]$	je hodnota jasu j-tého pixelu z okolí pixelu (x, y)

$$\delta[k, l] = \begin{cases} 1 & \text{pro} \quad k = 0 \wedge l = 0 \\ 0 & \text{pro} \quad k \neq 0 \vee l \neq 0 \end{cases}$$

Koeficient odlišnosti d dvou takovýchto histogramů H_1, H_2 je

$$d = \sum_{i=0}^{255} \sum_{j=0}^{7} |H_1[i, j] - H_2[i, j]| \tag{4.8}$$

Také byly zkoušeny kombinace histogramu jasu a *Local Binary Patterns*. Cílem bylo odstranit stejně jako v předchozím případě necitlivost *LBP* příznaků vůči jasu a kontrastu v obraze. Histogram jasu totiž dobře zachycuje jas a kontrast. Propojení bylo zkoušeno jako vážená funkce. Nejprve byla vypočítána rozdílnost histogramů *Local Binary Patterns*, poté rozdílnost histogramů jasu, a to vše bylo vhodným vážením převedeno na jedno číslo.

Jako další funkce určující rozdílnost dvou oblastí v obraze byla užita následující úprava *Local Binary Patterns*. Klasické *LBP* generuje pro každý pixel na základě jeho okolí osmibitové číslo, které přísluší na jednu určitou z 256 pozic *LBP* histogramu. Číslo na této pozici je pak inkrementováno o jedničku. Výsledný histogram tak popisuje, kolik pixelů ve sledované oblasti vykazuje okolí s daným *LBP* příznakem. V následující úpravě je místo počtu pixelů s daným příznakem sledován 16ti stupňový histogram jasu pixelů s daným *LBP* příznakem, přičemž při výskytu pixelu s určitou třídou světlosti je přičítána hrubá aproximace Gaussovy funkce. Cílem je dosáhnout vyšší podobnosti u histogramů, kde sledované pixely měly pouze mírné odlišnosti v jasu.

$$H[n,m] \quad = \quad \sum_{i=0}^{C-1} G[n - F_{LBP}[x_i, y_i], m - \lfloor P[x_i, y_i]/16 \rfloor] \tag{4.9}$$

kde

C	je počet pixelů ve zkoumané oblasti
$F_{LBP}[x,y]$	je hodnota *LBP* příznaku v pixelu (x, y)
x_i	je x-ová souřadnice pixelu, který má ve zkoumané oblasti index i
y_i	je y-ová souřadnice pixelu, který má ve zkoumané oblasti index i
$P[x,y]$	je hodnota jasu pixelu (x, y) v rozsahu 0 až 255

$$G[k,l] \quad = \begin{Bmatrix} 1 & \text{pro} & k = 0 \wedge l = -3 \\ 2 & \text{pro} & k = 0 \wedge l = -2 \\ 8 & \text{pro} & k = 0 \wedge l = -1 \\ 10 & \text{pro} & k = 0 \wedge l = 0 \\ 8 & \text{pro} & k = 0 \wedge l = 1 \\ 2 & \text{pro} & k = 0 \wedge l = 2 \\ 1 & \text{pro} & k = 0 \wedge l = 3 \\ 0 & \text{pro} & k \neq 0 \vee l < -3 \vee l > 3 \end{Bmatrix}$$

Koeficient odlišnosti d dvou takovýchto histogramů H_1, H_2 je

$$d \quad = \quad \sum_{i=0}^{255} \sum_{j=0}^{15} |H_1[i,j] - H_2[i,j]| \tag{4.10}$$

U všech funkcí využívajících *Local Binary Patterns* byla zkoušena i varianta uniformních *LBP*, která nezahrnuje osmibitové kombinace vyskytující se často jako výsledek šumu v obraze. Odpovídající rovnice pro výpočet odlišnosti d je v takovém případě třeba upravit tak, aby suma byla určována pouze přes patřičné *LBP* patterns v histogramech.

Funkce, která bude vykazovat nejlepší vlastnosti při práci s černobílým obrazem, bude následně užita 3x, jednou pro každý z *RGB* kanálů. Výsledná odlišnost bude váženým součtem odlišností v každém barevném kanálu.

4.2.2 Postup detekce

Pomocí dříve zmíněných funkcí, jejichž cílem je vyjádřit odlišnost určité oblasti v obraze vůči jiné oblasti v obraze, je možné hledat postup detekce výrazných míst ve zdrojové textuře. Základní myšlenkou je nalézt skupinu nebo skupiny oblastí, které vykazují nejvyšší odlišnost pomocí některé z funkcí. Postup by měl využívat vždy jednu funkci, případně vážený součet více funkcí, a na konkrétní funkci by měl být maximálně nezávislý. Takto bude možné funkce snadno zaměňovat a nalézt co nejlepší řešení. Výsledky pro dále zmíněné postupy lze nalézt v kapitole Výsledky (6).

Pro všechny dříve zmíněné funkce odlišnosti je smysluplné určovat průměrný histogram H_p. Pro jednorozměrné funkce je

$$H_p[n] = \frac{\sum_{i=0}^{C-1} H_i[n]}{C} \tag{4.11}$$

a pro vícerozměrné je

$$H_p[n,m] = \frac{\sum_{i=0}^{C-1} H_i[n,m]}{C}, \tag{4.12}$$

kde

C	je počet histogramů pro výpočet průměrného histogramu
H_i	je histogram, který má ve skupině histogramů pro výpočet průměrného histogramu index i

Jako první byl zkoušen postup, kdy je spočítán průměrný histogram přes všechny oblasti v obraze, a následně je hledáno N oblastí, které od takto určeného histogramu vykazují nejvyšší míru odlišnosti d.

Obrázek 4.7: různé texturní oblasti v textuře zamračené oblohy [7]

Zdrojová textura však často vykazuje několik oblastí, které vzájemně vykazují velkou odlišnost. Jako příklad lze uvést texturu oblačné oblohy, kde lze vidět bílé oblastí mraků,

modré oblasti oblohy a případně také přechody mraků v oblohu na jejich okrajích, jak ukazuje obrázek 4.7.

U takové zdrojové textury lze očekávat, že metoda vypočtu průměrného histogramu ze všech oblastí v obraze, kdy je určeno N nejodlišnějších oblastí, nebude dobře fungovat. Získaný průměrný histogram bude totiž mísit vizuálně velmi odlišné oblasti. Vhodnější by tedy bylo nalézt postup, kdy jsou ve zdrojové textuře identifikovány třídy vzájemně se podobajících oblastí, a z těch poté určovat průměrné histogramy. Místo hledání N oblastí nejodlišnějších vůči jedinému průměrnému histogramu, by byly hledány oblasti vykazující výrazné odlišnosti od histogramů všech tříd vzájemně podobných okolí.

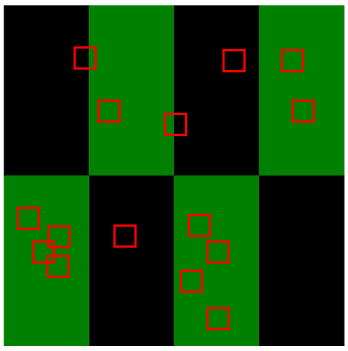
Obrázek 4.8: nesouvislé oblasti v textuře kamenné zdi, které vykazují podobnost [7]

Při sestavování skupin podobných oblastí ve zdrojové textuře je přístup pomocí metod segmentace obrazu problematický, protože ty obvykle předpokládají, že výsledné segmenty jsou v původním obraze souvislé. V případě textury však mohou oblasti, které přísluší do jediné třídy podobnosti, být rozmístěny v obraze zcela nesouvisle. Příkladem je textura kamenné zdi, kde např. jednu třídu tvoří středy cihel, jinou její hrany, jinou rohy a podobně, jak ukazuje obrázek 4.8.

Jednotlivé histogramy oblastí v obraze lze chápat jako body v N-rozměrném prostoru, jejichž vzdálenost d je definována pomocí některé z dříve zmíněných funkcí pro výpočet odlišnosti histogramů dvou oblastí v obraze. Takto je možné problém nalezení skupin vzájemně se podobajících okolí v obraze transformovat na problém segmentace bodů v N-rozměrném prostoru.

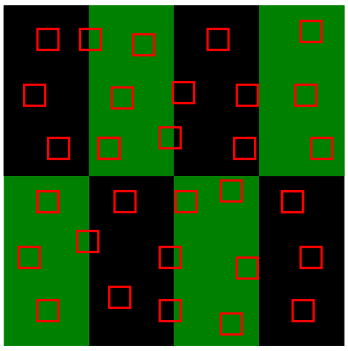
Body N-rozměrném prostoru s definovanou vzájemnou vzdáleností je možné segmentovat následujícím algoritmem. Každý bod je ve výchozím stavu považován za samostatnou skupinu bodů. V každém kroku je pak nalezena dvojice bodů, které mají v množině nejmenší vzájemnou vzdálenost, a zároveň přísluší do dvou různých skupin. Skupiny těchto dvou bodů jsou nadále považovány za jedinou skupinu. Po určitém počtu kroků se pak dosáhne optimálně segmentované množiny bodů. Tento algoritmus má však kubickou časovou složitost $O(N^3)$, kde N je počet segmentovaných bodů. Je proto obvykle v praxi obtížně použitelný, protože počet segmentovaných bodů bývá vysoký a je-li umocněný na 3, je doba výpočtu nepřijatelná. Proto se využívají metody segmentace s nižší časovou složitostí, které ale neposkytují ideálně segmentovaný výsledek. Zdrojová textura jako obraz však vykazuje vlastnosti, které umožňují počet bodů N výrazně redukovat. Opakují se v ní totiž často podobné oblasti a cílem je přiřadit je do tříd podobnosti. Lze proto očekávat, že průměrné histogramy tříd podobnosti počítané ze všech oblastí v obraze budou velmi podobné histo-

gramům získaným segmentací relativně malé množiny náhodně zvolených oblastí.



Obrázek 4.9: nevhodně rozmístěné oblasti pro reprezentaci textury, zelené oblasti zahrnuty výrazně častěji než černé

Pro co nejlepší pokrytí různých oblastí v obraze je však vhodné nevolit okolí zcela náhodně. Takto zvolená okolí by totiž nemusela dobře pokrýt celou plochu zdrojové textury a některá okolí byla nemusela být zahrnuta v dostatečné míře, jak ukazuje obrázek 4.9.

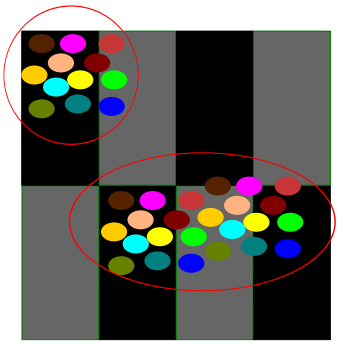


Obrázek 4.10: vhodně rozmístěné oblasti pro reprezentaci textury, zelené oblasti zahrnuty takřka stejně často jako černé

Lepší bude zvolit oblasti částečně náhodně, avšak tak, aby dobře pokrývaly plochu zdrojové textury, jak ukazuje obrázek 4.10.

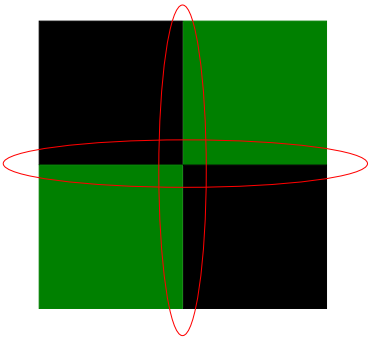
Počet takto získaných oblastí je relativně nízký, přestože průměrné histogramy ze získaných tříd podobnosti jsou velmi podobné histogramům získaným segmentací celé zdrojové textury. Je proto možné použít metodu segmentace s kubickou složitostí, která poskytuje na segmentované množině optimální výsledky.

Ve zdrojové textuře často dochází k tomu, že určité části textury, subjektivně vnímané jako jednolité, vykazují při analýze detekční funkcí větší vzájemnou odlišnost dílčích oblastí než jiné části textury, které jsou subjektivně vnímány rovněž jako jednolité. Při segmentaci pak dochází k nežádoucímu sjednocování velkých skupin oblastí, které mají výraznou vzájemnou odlišnost, pokud jiné, byť malé skupiny vykazují větší odlišnosti. Pokud však během algoritmu segmentace vznikly dvě již relativně velké skupiny, není už žádoucí je sjednotit, protože by se tak opět projevil negativní efekt průměrného histogramu počítaného přes značně odlišné části obrazu. Místo toho je vhodné v případě, že skupina dosáhne jisté kritické velikosti, neumožnit její sjednocení s jinou skupinou, která má také už kritickou velikost. Velké celistvé oblasti ve zdrojové textuře tak nebudou spojeny v jedinou pouze



Obrázek 4.11: Zakroužkované regiony v textuře by vytvořily velké množství malých skupin, které se velmi liší, a způsobily by sjednocení černé a šedé oblasti textury v jedinou.

proto, že jiná velká část textury vykazuje oblasti, které jsou si velmi málo podobné. Skupiny, které dosáhly kritické velikosti, tak mohou pouze růst, ale už nemohou být sjednoceny s jinými. Podstatu problému demonstruje obrázek 4.11.



Obrázek 4.12: Zakroužkované hraniční regiony budou pravděpodobně detekovány jako výrazné oblasti, protože vznikou průměrné histogramy pro zcela černou a zcela zelenou oblast.

Pokud sjednotíme vzájemně podobné oblasti do malého počtu skupin a budeme hledat maximálně odlišné oblasti, je pravděpodobné, že nalezneme oblasti na hranici částí textury, ze kterých jsou skupiny sestaveny. Problém demonstruje obrázek 4.12. Řešením je sestavit histogramy těchto hranic dodatečně a přidat je jako detekované skupiny do další analýzy výrazných oblastí. Vzhledem k tomu, že histogramy všech dříve zmíněných funkcí odlišnosti oblastí nejsou ve větším měřítku závislé na pozici pixelů, je možné histogramy hranic počítat jako průměrný histogram z průměrných histogramů jednotlivých nalezených skupin. Je možné sestavit dvojice a trojice z dříve detekovaných průměrných histogramů, a pak opět průměrováním dvojice či trojice sestavit hraniční histogramy. Pro lepší popis přechodu jedné skupiny v druhou je vhodné z dvojic sestavit více než jeden hraniční histogram pomocí váženého průměrováni histogramů.

Pokud jsou ke zdrojové textuře k dispozici průměrné histogramy jednotlivých skupin vzájemně podobných oblastí, lze přistoupit k detekci oblastí, které vykazují nejvyšší odlišnost. Pro každou oblast ve zdrojové textuře je spočítán histogram, a ten se porovná se všemi průměrnými histogramy dříve nalezených skupin. U nejmenšího z koeficientů odlišnosti se určí, zda je větší než N doposud největších koeficientů odlišnosti, a pokud ano, je v těchto koeficientech nejmenší nahrazen takovýmto novým. U každého koeficientu je rovněž nutné

udržovat informaci o tom, ke které oblasti přísluší. Jakmile jsou výše zmíněným postupem analyzovány všechny oblasti ve zdrojové textuře, je k dispozici informace o N nejodlišnějších oblastech ve zdrojové textuře. Oblasti jsou zvoleny čtvercové a vzájemně se překrývající. Je tedy pravděpodobné, že subjektivně výrazná místa v obrázku budou pokryta velkým počtem vzájemně se překrývajících detekovaných oblastí, a koeficient odlišnosti bude záviset na tom, jak velký obsah z výrazného místa oblast obsahuje.

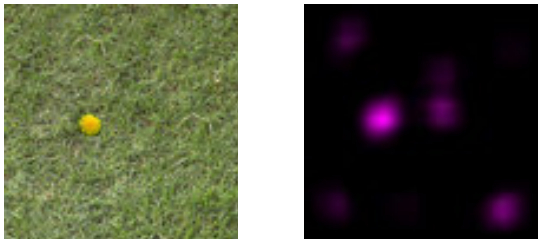
N takto nalezených nejodlišnějších oblastí je užito k určení dvoudimenzionální funkce odlišnosti oblasti v obraze F, a to následujícím způsobem. Inicializuje se dvoudimenzionální pole o výšce a šířce zdrojové textury na hodnoty 0. Následně je na pozici pixelů každé z N detekovaných oblastí přičten dříve určený koeficient odlišnosti.

$$F[n,m] = \sum_{i=0}^{N-1} d_i \cdot P_i[n,m] \tag{4.13}$$

kde

N	je počet nejodlišnějších detekovaných oblastí
d_i	je koeficient odlišnosti oblasti s indexem i

$$P_i[n,m] = \begin{cases} 1 & \text{pokud pozice [n, m] patří do oblasti s indexem i} \\ 0 & \text{pokud pozice [n, m] nepatří do oblasti s indexem i} \end{cases}$$



Obrázek 4.13: funkce odlišnosti oblastí pro texturu trávy s pampeliškou

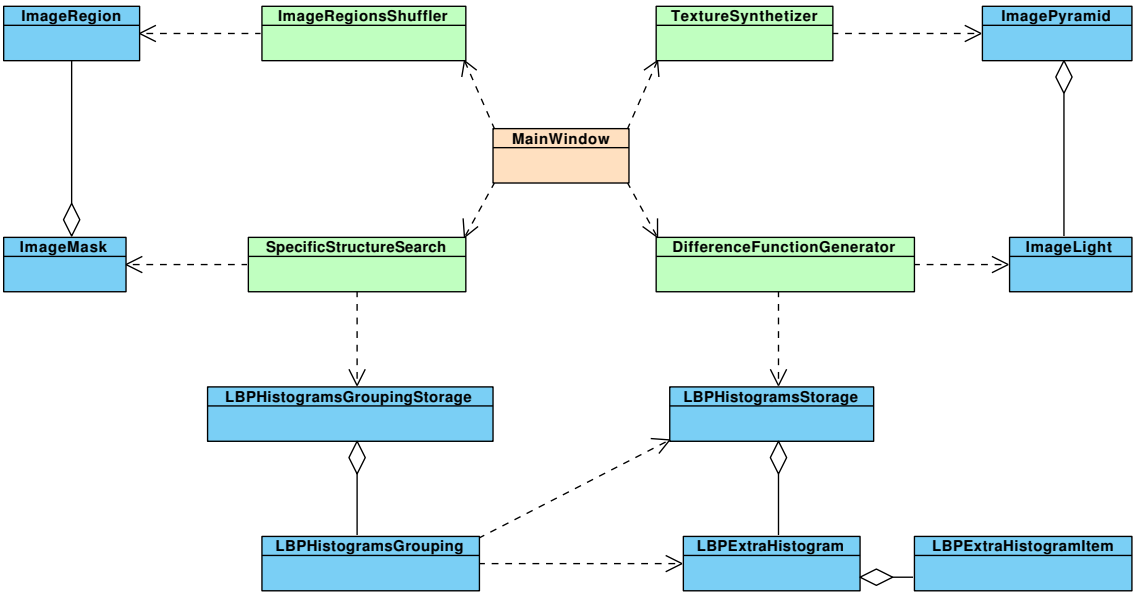
Příklad určené funkce odlišnosti oblastí v obraze je patrný na obrázku 5.1.

Kapitola 5

Implementace

Utilita byla implementována v jazyce C++ v operačním systému Linux. K vytvoření grafického uživatelského rozhraní byl použit toolkit Qt. Většina požitých algoritmů není k dispozici v již existujících knihovnách a bylo je nutné kompletně implementovat. Na základní operace načítání, ukládání a případně zmenšování obrazu pro obrazové pyramidy byla použita knihovna OpenCV - Open Computer Vision Library. Určitým problémem při implementaci byla relativně vysoká výpočetní náročnost jak algoritmů syntézy textur, tak algoritmů detekce výrazných oblastí.

Implementace je objektová a nejdůležitější třídy ukazuje následující diagram 5.1.



Obrázek 5.1: Diagram nejdůležitějších tříd a vztahů

Kapitola 6

Výsledky

6.1 Detekční funkce

První testovanou funkcí byl klasický histogram jasu oblasti v obraze. Jak lze očekávat, vykazoval dobré výsledky pro zvláštnosti ve zdrojové textuře tam, kde se oblasti výrazně lišily jasem nebo kontrastem. Ukázalo se však, že velmi častý je i případ, kdy se výrazný region jasem, kontrastem a rozložením jednotlivých skupin jasu liší pouze málo, avšak odlišnost spočívá v jiné struktuře dvou porovnávaných oblastí. Tuto třídu zvláštností ve zdrojové textuře nebyl barevný histogram schopen vůbec zachytit.

Pro statistické zachycení struktury oblasti v obraze byly použity *Local Binary Patterns*. Vzhledem k tomu, že se jedná o metodu přímo určenou na popis textury, byly u této funkce očekávány dobré výsledky. Ukázalo se však, že velká odolnost *Local Binary Patterns* proti změně jasu a kontrastu v obraze je pro danou aplikaci nevhodná. Funkce vykazovala dobré výsledky u výrazných oblastí ve zdrojové textuře, pokud se výrazně odlišovaly strukturou, avšak téměř zcela ignorovala změny jasu a kontrastu. Tato forma odlišnosti výrazných oblastí byla však u zdrojové textury velmi častá. Stejně jako histogramy jasu oblastí v obraze funkce fungovala pouze pro určitou třídu odlišností.

Vzhledem k tomu, že *Local Binary Patterns* dobře zachycovaly ve zdrojové textuře rozdíly oblastí podle jejich struktury a histogram jasu zase dobře zachycoval změny jasu a kontrastu, byly učiněny pokusy jak tyto dva koncepty spojit v jeden, který by sdílel silné stránky obou.

Vážený součet koeficientů odlišnosti obou funkcí však překvapivě nefungoval příliš dobře. V určitých případech byly výsledky horší než při použití každé z dílčích funkcí zvlášť. Jako částečné vysvětlení tohoto jevu lze uvést, že při analýze metodou popsanou v 4.2.2 byly při použití každé z dílčích funkcí zvlášť generovány natolik odlišné skupiny podobných oblastí v obraze, že při spojení těchto funkcí se jejich koncepty začaly spíše potírat nežli vzájemně podporovat.

Další funkce odlišnosti okolí se tedy pokoušejí rozšířit *Local Binary Patterns*, které vykazovaly slibné počáteční výsledky, aby bylo dosaženo citlivosti na změny jasu a kontrastu v obraze.

Funkce sledující součet jasu pixelů s okolím s konkrétním *LBP* příznakem podle rovnice (4.5) vykazovala lepší výsledky než kterákoliv z dříve zmíněných, ale některých zdrojových texturách fungovala nepříliš dobře. Funkce sčítá hodnoty jasu ve středovém pixelu, a tím zapříčiňuje určité nežádoucí průměrování. Pokud mají pixely ve sledované oblasti okolí se stejným *LBP* příznakem, a zároveň se výrazně liší v jasu středového pixelu, je jas zprůměrován z hlediska výpočtu odlišnosti histogramů. Pro konkrétní *LBP* příznak

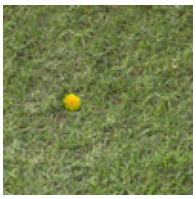
tak pixely vzájemně velmi odlišné jasem generují velmi podobný histogram jako pixely s průměrem těchto hodnot jasu. Tento efekt způsobuje výrazné selhávání metody detekce pro určitou třídu výrazných oblastí ve zdrojové textuře.

Dále byla zkoušena funkce podle rovnice (4.7), která sleduje relativní vztahy mezi jasem středového pixelu a jasem pixelu v jeho okolí pro daný *LBP* příznak. Stejně jako předchozí se funkce projevovala slibně z hlediska výsledků, avšak ve specifickém avšak relativně často se vyskytujícím případě selhávala. Protože funkce sleduje relativní vztahy mezi pixely jako součet rozdílů jasu, docházelo k nežádoucímu jevu, kdy kontrastnější oblasti jsou vzájemně vyhodnoceny jako odlišnější nežli oblasti méně kontrastní, a to i tam, kde to bylo subjektivně vnímáno zcela jinak. Také se ukázalo, že funkce je málo citlivá k celkovému jasu oblastí, zatímco přeceňuje rozdíly kontrastu.

Funkce podle rovnice (4.9) se snaží odstranit dříve zjištěné problémy. Je sledován histogram jasu pro pixely s daným *LBP* přínakem. Tato funkce vykazovala jednoznačně nejslibnější výsledky, a byla použitelná pro nejširší škálu různých druhů zvláštností ve zdrojové textuře. Byla proto vybrána pro finální verzi aplikace, a všechny následující výsledky už vycházejí z použití této funkce. Konkrétně byla užita její drobná úprava vycházející z uniformních *Local Binary Patterns*, a některé *LBP* osmibitové kombinace tak nejsou sledovány a užívány pro výpočet rozdílu histogramů. Ve finální podobě je navíc počítána zvlášť pro každý z *RGB* kanálů, a výsledný koeficient odlišnosti d je váženým součtem koeficientů odlišnosti pro jednotlivé kanály. Cílem je zachycovat také barevné odlišnosti, které se dobře neprojeví v obraze v odstínech šedi.

6.2 Výsledná metoda syntézy

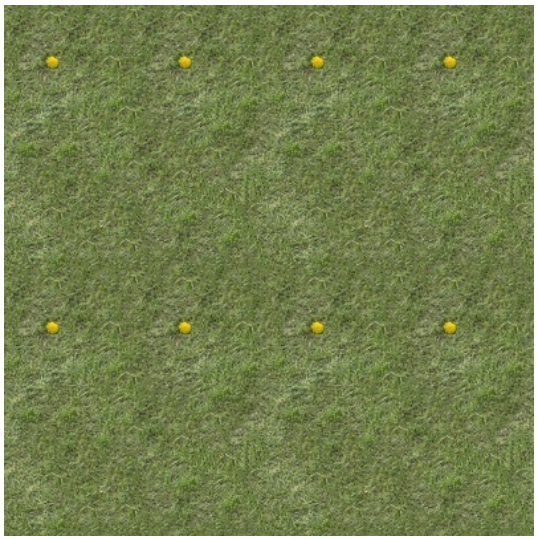
Následují výsledky syntézy v obrazové podobě. Pokud je pro zdrojovou texturu k dispozici výsledek původní metody syntézy autorů Li-Yi Weie a Mrca Levoye, je jejich výsledek pro srovnání rovněž zahrnut.



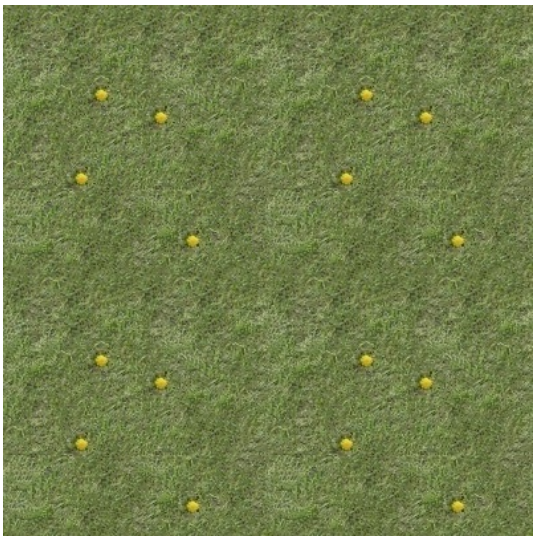
(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí



(d) 400x400 bez inicializace detekcí výrazných oblastí - stranová návaznost



(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek 6.1:

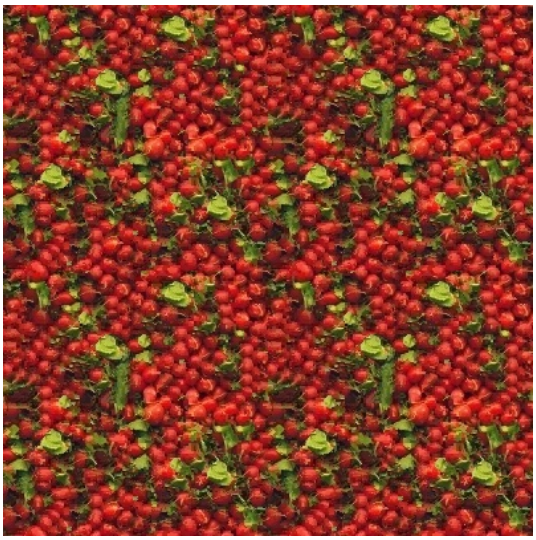
V obrázku 6.1b, který byl generován bez použití inicializace pomocí detekce výrazných oblastí, je patrný výskyt výrazné pampelišky pouze v horní části obrázku. Jak je vidět na 6.1d, tak při stranově navazujícím skládání textury vzníka horizontální pruh pampelišek, který není žádoucí. Pokud je však použita inicializace jako v 6.1c, je oblast obsahující pampelišku detekována a lépe rozmístěna po ploše výsledné syntetizované textury. Míru náhodností rozmístění detekovaných regionů lze nastavit a regulovat tak pravidelnost rozmístění pampelišek ve výsledné textuře.

(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí

(d) 400x400 bez inicializace detekcí výrazných oblastí - stranová návaznost



(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek 6.2:

Obrázek 6.3: výsledek autorů Li-Yi Wei, Marc Levoy [8]

V syntetizované textuře ředkviček je tam, kde nebyla použita inicializace pomocí detekční funkce (6.2d), patrné opakování jasně zeleného lístečku v jedné linii, což působí rušivě. Pomocí detekční funkce byl tento lísteček identifikován a lépe rozložen po ploše syntetizované textury (6.2e). Rušivý efekt opakující se linie tak byl výrazně zmírněn.

(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí

(d) 400x400 bez inicializace detekcí výrazných oblastí - stranová návaznost

(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek 6.4:

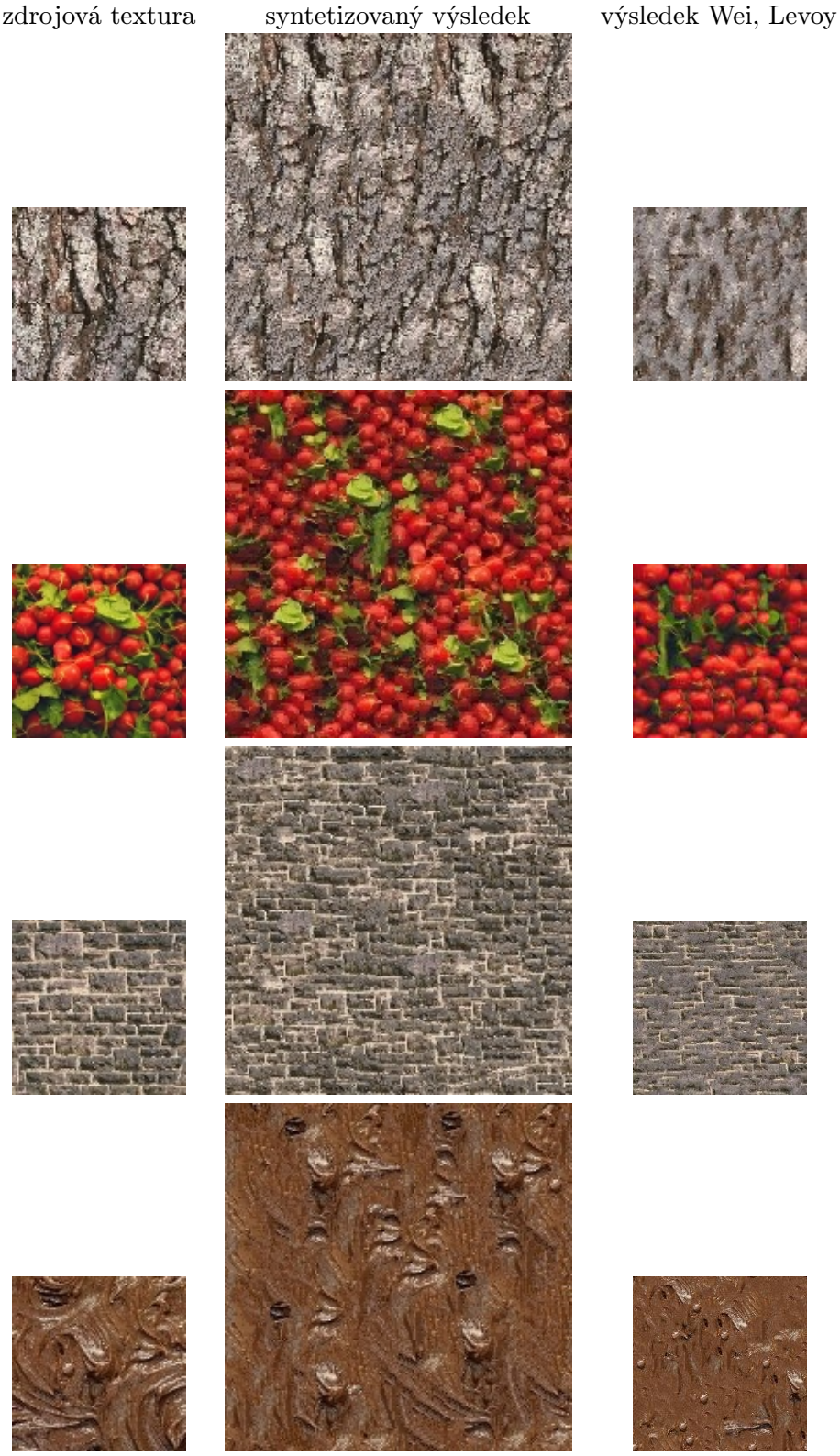
Obrázek 6.5: výsledek autorů Li-Yi Wei, Marc Levoy [8]

Na obrázku 6.4d je jasně patrné opakování výrazného výrůstku ve struktuře kůry v horizontální linii. Díky detekcí výrazných regionů se podařilo tuto oblast nalézt, a lépe zajistit její zahrnutí do výsledné textury 6.4e.

Textury v této kapitole mají rozměry 200×200 pixelů a byly generovány ze zdrojové

textury 100×100 pixelů. Samotný proces generování zabírá v průměru 750s na jednom jádře 2.1GHz procesoru. Detekce výrazných oblastí trvala v průměru 49s. Doba výpočtu algoritmu syntézy textury roste přibližně kvadraticky se stranou jak zdrojové tak cílové textury.

Detekce výrazných oblastí ve zdrojové textuře je schopna výrazně zmírnit následky pravidelného rozmístění takovýchto oblastí v cílové textuře. Je možné navíc regulovat míru toho, jak náhodně budou oblasti ve výsledné textuře rozmístěny. V případech, kdy obrázek neobsahuje příliš výrazné regiony, přináší detekční funkce často rovněž pozitivní výsledek díky tomu, že inicializace lépe udržuje strukturu zdrojové textury, co se týče rozmístění jednotlivých částí. Se vzdáleností od pevné inicializace má totiž metoda tendenci mírně degradovat a zvyšovat náhodnost generovaného výsledku. Další příklady syntetizovaných textur je možné najít v příloze.



Obrázek 6.6: srovnání výsledků syntézy s původní metodou Li-Yi Weie a Marca Lewoye

Kapitola 7

Závěr

Cílem této bakalářské práce bylo upravit některou existující metodu syntézy textur tak, aby bylo dosaženo co nejlepších výsledků co se týče vzhledu výsledné textury. Hlavní myšlenkou bylo upravit postup syntézy tak, aby bral v úvahu výrazné oblasti ve zdrojové textuře a ty tak byly lépe zahrnuty do výsledné syntetizované textury. Pro subjektivní hodnocení kvality výsledku jsou totiž tyto oblasti zásadní.

Výsledky získané zde popsanou metodou byly ve většině případů lepší než výsledky syntetizované pomocí algoritmu autorů Li-Yi Weie a Marca Levoye, který posloužil jako základ algoritmů syntézy. Rozšíření pomocí detekce výrazných oblastí ve zdrojové textuře se ukázalo jako použitelné a ve většině případů zvyšovalo kvalitu výsledku.

Metoda nejlépe funguje pro textury, v nichž se nevyskytují objekty s jasně definovaným tvarem. Z principu udržuje pouze návaznost částí textury na sebe, ale objekty samotné pochopitelně nevnímá.

Z hlediska další práce by bylo možné výrazně urychlit algoritmy syntézy, které jsou v současné podobě příliš pomalé pro praktické nasazení. Také je možné dále zkoumat a zlepšit detekční funkci, což by patrně vedlo k dalšímu zlepšení syntetizovaných výsledků.

Literatura

[1] Arno, V. K.; Schödl, A.; Essa, I.; aj.: Graphcut Textures: Image and Video Synthesis Using Graph Cuts. 2003.

[2] Efros, A.; Leung, T.: Texture Synthesis by Non-parametric Sampling. In *In International Conference on Computer Vision*, 1999, s. 1033–1038.

[3] Efros, A. A.; Freeman, W. T.: Image Quilting for Texture Synthesis and Transfer. 2001.

[4] Paget, R.; Longstaff, I. D.; Member, S.: Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field. 1998.

[5] Pietikäinen, M.: Image Analysis with Local Binary Patterns. In *Image Analysis*, 2005, s. 115–118.

[6] Wei, L.-Y.; Levoy, M.: Fast Texture Synthesis using Tree-structured Vector Quantization. 2000, s. 479–488.

[7] WWW stránky: Texture Synthesis : VisTex Textures [online, přístupné 13.5. 2010]. `http://graphics.stanford.edu/projects/texture/demo/synthesis_VisTex_192.html`.

[8] WWW stránky: Texture Synthesis Examples [online, přístupné 13.5. 2010]. `http://graphics.stanford.edu/projects/texture/demo/`.

Dodatek A

Syntetizované textury

(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí

(d) 400x400 bez inicializace detekcí oblastí - stranová návaznost

(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek A.1:

Výsledek syntézy z textury zelených listů působí podobně v inicializované i neinicializované podobě, avšak je patrné, že díky inicializaci nevymizel výrazný stín pod listy. Díky inicializaci se také výsledek více drží struktury zdrojové textury a působí poněkud jasněji.

(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí

(d) 400x400 bez inicializace detekcí výrazných oblastí - stranová návaznost

(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek A.2:

V této textuře mořského příboje je patrné, že díky inicializaci byl zmírněn negativni efekt, kdy se výrazná modrá skvrna ze středu zdrojové textury opakuje ve dvou horizontálních liniích v případě obrázku stranové návaznosti.

(a) 100x100 zdrojová textura

(b) 200x200 bez inicializace detekcí výrazných oblastí

(c) 200x200 s inicializací detekcí výrazných oblastí

(d) 400x400 bez inicializace detekcí výrazných oblastí - stranová návaznost

(e) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek A.3:

Textura trávy s jetelem ukazuje, jak bylo díky inicializaci o něco lépe zachováno plošné rozdělení jednotlivých květů jetele ve výsledné syntetizované textuře.

(a) 100x100 zdrojová textura

(b) 200x200 s inicializací detekcí výrazných oblastí

(c) Li-Yi Wei, Marc Lewoy

(d) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek A.4:

(a) 100x100 zdrojová textura

(b) 200x200 s inicializací detekcí výrazných oblastí

(c) Li-Yi Wei, Marc Lewoy

(d) 400x400 s inicializací detekcí výrazných oblastí - stranová návaznost

Obrázek A.5: